LA-UR 97-480

Modifying Edges of a Network to Obtain Short Subgroups

Authors: K.U. Drangmeister, S.O. Krumke, M.V. Marathe,
H. Noltemeier, S.S. Ravi

September 1996

LOS ALAMOS
NATIONAL LABORATORY

Modifying Edges of a Network to Obtain Short Subgraphs

K. U. Drangmeister [1] S. O. Krumke [1] M. V. Marathe [2]
H. Noltemeier [1] S. S. Ravi [3]

September 17, 1996

Abstract

This paper considers problems of the following type: We are given an edge weighted graph $G = (V, E)$. It is assumed that each edge e of the given network has an associated function c_e that specifies the cost of shortening the edge by a given amount and that there is a budget B on the total reduction cost. The goal is to develop a reduction strategy satisfying the budget constraint so that the total length of a minimum spanning tree in the modified network is the smallest possible over all reduction strategies that obey the budget constraint.

We show that in general the problem of computing an optimal reduction strategy for modifying the network as above is NP-hard even for simple classes of graphs and linear functions c_e. We present the first polynomial time approximation algorithms for the problem, where the cost functions c_e are allowed to be taken from a broad class of functions. We also present improved approximation algorithms for the class of treewidth-bounded graphs when the cost functions are linear. Our results can be extended to obtain approximation algorithms for more general network design problems such as those considered in [GW92, GGP$^+$94].

Keywords: NP-hardness, Approximation Algorithms, Network Design, Spanning-Tree.

[1]Department of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany. Email: {drangmei,krumke,noltemei}@informatik.uni-wuerzburg.de.

[2]Los Alamos National Laboratory, P.O. Box 1663, MS K990, Los Alamos, NM 87545, USA. Email: madhav@c3.lanl.gov. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

[3]Department of Computer Science, University at Albany – SUNY, Albany, NY 12222, USA. Email: ravi@cs.albany.edu.

1 Introduction

We study network design problems where the goal is to find optimal improvement strategies for modifying a given network. Such problems arise in diverse areas including design of high speed communication networks [KJ83], video on demand [KPP93], teleconferencing [KPP92a] and VLSI design [CKR+92, ZPD94]. For example, consider the following scenario that arises in the cost/benefit analysis for improving communication networks. A large communication company is approached by a client with the requirement to interconnect a set of cities housing the client's offices, e.g. banks with a high transaction rate between the sites. The company has a list of feasible links that it can use to construct a network to connect these cities. Each link has a construction cost associated with it. One of the main concerns of the client is to build a communication network of minimum cost. This is the ubiquitous minimum spanning tree problem. With the advent of optical communication technology, the client would like to upgrade the communication network and has allocated a certain budget to do so. In general, there is a cost for improving each link in the existing network by a unit amount. The goal is to design a strategy to upgrade the links of the network so that the total cost of upgrading the links is no more than the allocated budget, and the cost of a minimum spanning tree for the upgraded network is the least over all the possible improvements of the network satisfying the budget constraint.

The problem stated above is an example of an *edge based network improvement problem*. In this paper we focus on such problems for undirected graphs. The nodes of the graph represent the set of sites (offices). A cost function specifies the cost of improving an edge by a given amount. For a given budget B and a class of subgraphs $\mathcal{S}$, the goal is to find a reduction strategy such that the total cost of reduction is at most B and the minimum cost subgraph $S \in \mathcal{S}$ (with respect to some measure $\mathcal{M}$) under the upgraded costs is the best over all possible reduction strategies which obey the budget constraint. In this paper, we restrict our attention to cases in which $\mathcal{M}$ is the total cost or the diameter of the subgraph. The class of subgraphs $\mathcal{S}$ considered includes spanning trees, Steiner trees, generalized Steiner forests, etc. A main contribution of this paper is a general technique for obtaining the first polynomial time approximation algorithms for a large class of edge based network improvement problems.

The remainder of the paper is organized as follows. Section 2 contains basic definitions and formal statements of the problems considered in this paper. It also discusses a framework for evaluating approximation algorithms. Section 3 summarizes the results in the paper. Section 4 discusses related work. In Section 5, we briefly discuss the structure of optimal solutions. Section 6 contains the complexity results for solving the problems optimally. Section 7 contains our approximation algorithms for general graphs. Section 8 presents a faster approximation algorithm for linear cost functions. Section 9 discusses the extensions of these algorithms to other subgraph classes and Section 10 discusses improved approximations for the class of treewidth bounded graphs. Finally, Section 11 contains directions for future research.

2 Problem Formulation and Approximation Framework

Let $G = (V, E)$ be an undirected graph. Associated with each edge $e \in E$, there are nonnegative values as follows: $\ell(e)$ denotes the *length* or the *weight* of the edge e and $\ell^{\min}(e)$ denotes the *minimum length* to which the edge e can be reduced. Consequently, we assume throughout the presentation that $\ell^{\min}(e) \leq \ell(e)$. The nonnegative *cost function* c_e indicates how expensive it is to reduce the length of e by a certain amount. We assume without loss of generality that $c_e(0) = 0$ for all edges $e \in E$.[4]

A *reduction strategy* (or simply *reduction*) on the edges of G specifies how to reduce the ℓ-length of each edge e to a value in the range $[\ell^{\min}(e), \ell(e)]$. Given a budget B, we define a *feasible reduction* to be a nonnegative function r defined on E with the following properties: For all edges $e \in E$, $\ell(e) - r(e) \geq \ell^{\min}(e)$ and $\sum_{e \in E} c_e(r(e)) \leq B$. If r is a (feasible) reduction, we can consider the graph G with edge weights given by the "reduced lengths", namely $(\ell - r)(e) := \ell(e) - r(e)$ $(e \in E)$.

Let $\mathcal{S}$ be a subgraph class and let $S \in \mathcal{S}$ be a subgraph of G. The *total length* of S under the weight function ℓ, denoted by $\ell(S)$, is defined to be the sum of the lengths of the edges in S. We denote a minimum total length subgraph in $\mathcal{S}$ with respect to the weight function ℓ by $\mathcal{S}^*_G(\ell)$. Similarly, if r is a (feasible) reduction in G then $\mathcal{S}^*_G(\ell - r)$ denotes a minimum total length subgraph with respect to the reduced lengths $\ell(e) - r(e)$ $(e \in E)$. We omit the graph G in the subscript whenever such an omission does not cause any ambiguity. In what follows we will often use the same symbol for a subgraph and its cost and the intended meaning will be clear from the context.

For some versions of the problems discussed in the sequel, we impose some additional constraints on permissible reductions. Thus, we obtain the following three cases:

1. For each edge e, the reduction must either shorten the length of the edge to $\ell^{\min}(e)$ or leave the length unchanged. Formally, we require each (feasible) reduction to satisfy the condition $r(e) \in \{0, \ell(e) - \ell^{\min}(e)\}$ for all $e \in E$. These reductions will be referred to as *0/1-reductions*.

   Note that another way to view a 0/1-reduction r is to use it to model the *insertion of alternative edges* to the graph G, with the reduction of the edge e corresponding to the insertion of a new edge $\hat{e}$ parallel to e with $\ell(\hat{e}) = \ell^{\min}(e)$.

2. The reduction r must be an integer valued function; i.e., for each edge e, $r(e)$ must be an integer in $\{0, 1, \ldots, \ell(e) - \ell^{\min}(e)\}$. We denote this type of reductions by *I-reductions* ("integer reductions").

3. The third case is the least restricted one. Here we allow a reduction r to take on rational values; i.e., for each edge e, the reduction can be a rational value

[4] Any reduction will incur a minimum cost of $\sum_{e \in E} c_e(0)$ and we can subtract this sum from the budget B in advance.

in $[0, \ell(e) - \ell^{\min}(e)]$. We refer to these reductions as *R-reductions* ("rational reductions").

The reader may wonder why it is necessary to look at the various types of reduction strategies. As the subsequent sections show, for several problems considered here, the complexity of obtaining an optimal solution depends on the type of reduction strategy used. In contrast, the approximation algorithms we devise generally work for any of the three variants simultaneously.

We are now ready to formulate the problems studied in this paper. Our formulation is based on the work of [MRS+95]. A generic edge based network improvement problem $(f_1, f_2, \mathcal{S})$, is defined by identifying two minimization objectives, f_1 and f_2, from a set of possible objectives, and specifying a membership requirement in a class of subgraphs, $\mathcal{S}$. The problem specifies a budget value B on the first objective, f_1, under c_e cost function, and seeks to find a subgraph $S \in \mathcal{S}$ such that the cost of S is a minimum with respect to the second objective, f_2, under the modified cost function $\ell - r$. The cost of upgrading the network as measured by f_1 under c_e should be no more than B. For the budgeted objective f_1, we focus on the total cost of upgrading the network. As mentioned earlier, upgrading of edges can be carried out by a reduction r that is 0/1 or integral or rational. We use these three types to further classify f_1. Thus $f_1 \in \{$ 0/1-Upgrade-Total Cost, I-Upgrade-Total Cost, R-Upgrade-Total Cost $\}$. For the minimization objective f_2, we consider the *total cost* of all the edges in the subgraph. Finally, for the problems considered here $\mathcal{S} \in \{$ Spanning Tree, Steiner Tree, Generalized Steiner Tree, $\}$, etc.

For example, the improvement problem for obtaining a spanning tree of small length described in the earlier sections is the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem. In this problem, the goal is to find a reduction r of cost at most B such that $\mathrm{MST}_G(\ell - r)$ has the least possible value. Similarly, the goal of the (0/1-Upgrade-Total-Cost, Total Cost, Steiner Tree) problem is to find a shortest Steiner tree in the modified network under 0/1-reductions that obeys the budget constraint.

Most of the network improvement problems considered in this paper are NP-hard. In fact, for several problems (e.g. (0/1-Upgrade-Total-Cost, Total Cost, Steiner Tree)) we show that it is hard to find a solution that is near-optimal with respect to the objective, if the solution is required to satisfy the budget constraint. Given these hardness results, we focus on finding efficient approximation algorithms that guarantee a solution which is approximate in terms of both the budget and the objective function. We first discuss a measure to evaluate approximation algorithms for such network improvement problems.

Definition 2.1 *Let $\alpha, \beta \geq 1$ be constants. We say that an algorithm is an* (α, β)-approximation algorithm *for a $(f_1, f_2, \mathcal{S})$ problem, if for each instance, the algorithm returns a reduction r and a subgraph $S \in \mathcal{S}$ such that*

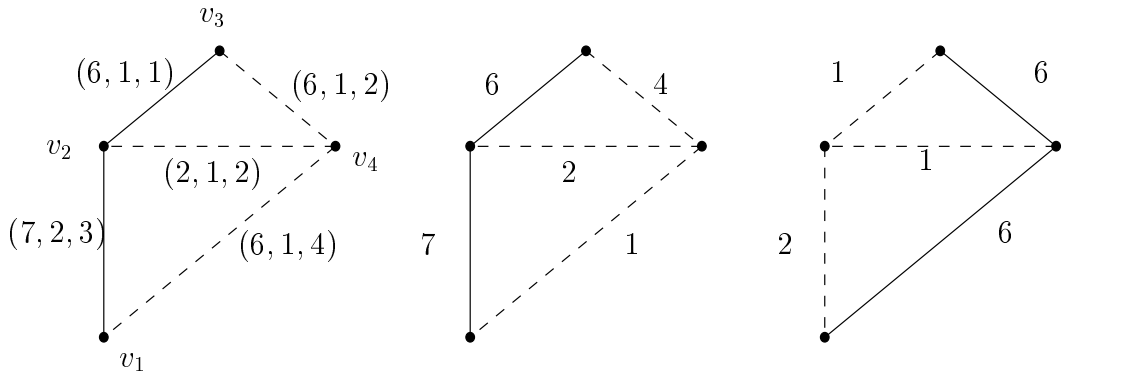
1. *The cost of the reduction (under f_1) is at most βB and*

2.

$$\frac{(\ell - r)(S)}{\mathcal{S}^*_G(\ell - r^*)} \quad \leq \quad \alpha, \tag{1}$$

where r^* denotes an optimal edge-reduction of cost at most B, $S^*_G(\ell - r^*)$ denotes the cost (under f_2) of an optimal subgraph in the network with cost function $\ell - r^*$ and $(\ell - r)(S)$ denotes the cost of the subgraph S with cost function $\ell - r$.

Example 2.2 Consider the graphs given in Figure 1. Figure 1(a) shows a graph G where each edge e is associated with the three values $(\ell(e), \ell^{\min}(e), c_e)$. The third parameter c_e represents the cost of reducing the length of the edge by a unit amount; i.e., the cost function on each edge in this simple example is linear and is given by $c_e(t) = c_e \cdot t$. The result of a modification of G is shown in Figure 1(b). The edges belonging to the minimum spanning tree are drawn as dashed lines. The modification corresponding to Figure 1(b) involves a cost of 24 and the weight of the resulting tree is 7. Figure 1(c) shows the graph with edge lengths resulting from a reduction that is optimal among all reductions of cost no more than 22. There, the weight of the spanning tree resulting from the reduction is 4. Thus, the reduction of Figure 1(b) is a $(7/4, 24/22)$-approximation to an optimal solution with budget 22.



(a) The original graph G (b) Modification of G with cost 24 (c) Optimal Modification for a budget of 22

Figure 1: An example of a graph modification via edge reductions.

3 Summary of Results

Here, for the first time in the literature, we study the complexity and approximability of several network improvement problems. We present both NP-hardness results and approximation algorithms with provable performance guarantees for the problems studied here. Wherever possible, we state the hardness results for the most restricted

versions of problems (e.g. for the spanning tree version) and the approximation results for the most general versions of problems (e.g. for the Steiner tree version). Also, our hardness results use simple linear cost functions while our approximation algorithms can handle a variety of cost functions.

1. We observe that the (0/1-Upgrade-Total-Cost, Total Cost, Spanning Tree) and (0/1-Upgrade-Total-Cost, Diameter, Spanning Tree) problems are NP-hard even when the underlying network is a tree and the reduction cost functions are linear. We show that the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem is NP-hard even when the underlying network is series-parallel and the reduction cost functions are linear.

2. For general graphs, we show that unless $\mathsf{P} = \mathsf{NP}$, for any $\rho > 1$, there is no polynomial time $(\rho, 1)$ or $(1, \rho)$ approximation algorithm for the problems (0/1-Upgrade-Total-Cost, Total Cost, Steiner Tree), (R-Upgrade-Total-Cost, Total Cost, Steiner Tree) and (I-Upgrade-Total-Cost, Total Cost, Steiner Tree).

3. For general graphs, we also show that unless $\mathsf{NP} \subseteq \mathsf{DTIME}(N^{\log\log N})$, for any $\varepsilon > 0$ and $0 < \mu < 1$, there is no $(11/10 - \varepsilon, \mu \ln B)$ approximation for the problems (0/1-Upgrade-Total-Cost, Diameter, Spanning Tree) and (I-Upgrade-Total-Cost, Diameter, Spanning Tree).

4. For general graphs, given any fixed $\gamma > 0$, we present a $(1 + 1/\gamma, 1 + \gamma)$ approximation algorithm for the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem. This algorithm can accommodate a variety of reduction cost functions. When the reduction cost functions are linear, we present an efficient implementation of the approximation algorithm using Megiddo's technique [Meg83]. For graphs of bounded treewidth, we give an improved approximation algorithm with a performance of $(1 + \varepsilon, 1 + \varepsilon)$ for any fixed $\varepsilon > 0$.

5. For general graphs, we present an $(\mathcal{O}(\log n), \mathcal{O}(\log n))$ approximation algorithm for the (R-Upgrade-Total-Cost, Diameter, Spanning Tree) problem.

Our approximation algorithm for (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) can be extended significantly. For example, using our ideas in conjunction with the results of Goemans et. al. [GGP+94], we can obtain similar approximation results for finding budget constrained minimum-cost generalized Steiner trees, minimum-cost k-edge connected subgraphs and other network design problems specified by weakly supermodular functions.

4 Comparison with Related Work

As far as we know, the problems considered in this paper have not been previously studied. Recently in an independent effort Frederickson and Solis-Oba [FSO96] considered the problem of increasing the weight of a minimum spanning tree in a graph

subject to a budget constraint where the cost functions are assumed to be linear in the weight increase. In contrast to the results presented here, they show that while the integral case is NP-hard, the rational case is solvable in polynomial time using tools from matriod theory. Berman [Ber92] considers the problem of shortening edges in a given tree to minimize its shortest path tree weight and shows that the problem can be solved in strongly polynomial time. Plesnik [Pl81] has shown that the budget-constrained minimum diameter problem (i.e., given a graph $G = (V, E)$ with a length $\ell(e)$ and cost $c(e)$ for each edge $e \in E$ and a cost budget B, select a subset E' of E so that the total cost of edges in E' is at most B and the diameter of the graph formed by E' is a minimum among all subsets satisfying the budget constraint) is NP-hard. He also shows that, if the budget constraint cannot be violated, then even approximating the diameter to within a factor of less than 2 is NP-hard. It can be seen that the problem considered by Plesnik is an edge-based network improvement problem under 0/1-reductions where there is a budget on the total upgrade cost and the goal is to improve the diameter of the network. The important difference between this problem and the (0/1-Upgrade-Total-Cost, Diameter, Spanning Tree) problem considered here is that the former problem does not require the subgraph induced by the modified edges to be a tree. Phillips [Phi93] studies the problem of finding an optimal strategy for reducing the capacity of the network so that the residual capacity in the modified network is minimized. The problems studied here and in [Phi93, Ber92] can be broadly classified as types of bicriteria problems. Recently, there has been substantial work on finding efficient approximation algorithms for a variety of bicriteria problems (see [KP95, Has92, MRS+95, RMR+93, Rav94, War92] and the references therein).

5 Structure of an Optimal Solution

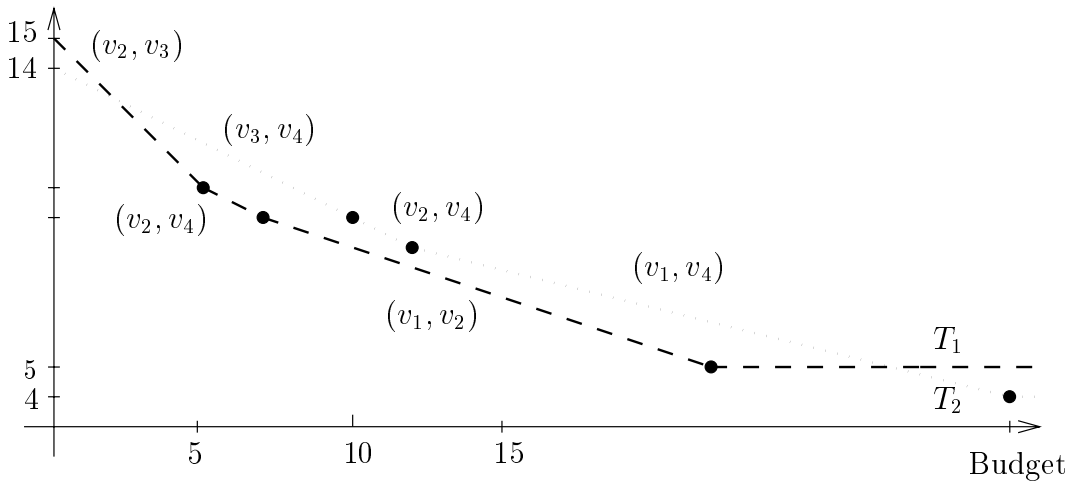


Figure 2: Remaining weight of the trees T_1 and T_2 as a function of the budget.

In this section we comment on the structure of optimal solutions to the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem for linear reduction costs on the edges, that is, $c_e(t) = c_e \cdot t$ for all $e \in E$ and some constants c_e. We also mention some special cases of the problem that can be solved in polynomial time.

First, suppose that the given budget B is zero. Then (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) reduces to the well known minimum spanning tree problem (with length function $\ell(e)$), and is known to be optimally solvable by classical algorithms (e.g. Kruskal's algorithm). Similarly, if $B = +\infty$ (i.e., there is no bound on the cost of upgrading the network), the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem again reduces to the minimum spanning tree problem but this time with edge-lengths given by $\ell^{\min}$.

Optimal solutions to (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) also exhibit some structure in the general case (i.e., $B \notin \{0, +\infty\}$). Any (feasible) reduction r induces a tree in a natural way, namely a minimum spanning tree T_r in the graph with the modified edge lengths. Observe that the quality of the solution produced via the reduction r depends solely on the weight of T_r, so all the cost incurred in upgrading edges not in T_r is wasted. Moreover, for any *fixed* tree T in G, the Greedy-strategy that successively reduces a cheapest available edge is an optimal reduction strategy. Thus, if we already knew a minimum spanning tree T_{r^*} corresponding to an optimal reduction r^*, we could solve (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) quite easily.

This observation also suggests a very simple exponential time algorithm for solving (R-Upgrade-Total-Cost, Total Cost, Spanning Tree): Enumerate all spanning trees in G, apply the above Greedy-strategy to each of them and then select the best solution. Unfortunately, a graph G with n nodes can have n^{n-2} different spanning trees.

We now discuss the *sensitivity* of optimal reduction strategies to changes in the given budget B. If we fix a spanning tree and plot the weight of that tree as a function of the money spent on it in a Greedy manner, we see that each piece corresponds to a budget range where one particular edge e is shortened. Thus it is easy to see that the piece has slope $-1/c_e$.

Figure 2 shows the plots corresponding to the tree T_1 consisting of the edges (v_2, v_3), (v_2, v_4), (v_1, v_2) and the tree T_2 consisting of the edges (v_3, v_4), (v_2, v_4) and (v_1, v_4) taken from the example graph of Figure 1. As can be seen from Figure 2, the plots for different trees can cross each other multiple times. If we plot the weights of all spanning trees on the same set of axes, the lower envelope gives the optimal remaining weight per budget. It is easy to see that the lower envelope can have an exponential number of linear pieces.

6 Hardness Results

In this section, we present NP-hardness and non-approximability results for the problems considered in this paper. We first show (Section 6.1) that several of these problems are NP-hard even for simple classes of graphs (trees and series-parallel graphs). Next, for general graphs, we strengthen our results and provide (Section 6.2) non-approximability results for several problems.

6.1 Results for Special Classes of Graphs

It is easy to see (c.f. Section 5) that when G is a tree and the cost functions c_e are all linear, (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) and (I-Upgrade-Total-Cost, Total Cost, Spanning Tree) problems can be solved optimally in polynomial time by a Greedy-type algorithm that simply keeps on reducing the length of the cheapest available edge. In contrast, as shown in the next proposition, the (0/1-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem is NP-hard even when G is a tree. The same construction also yields the NP-hardness of (0/1-Upgrade-Total-Cost, Diameter, Spanning Tree).

Proposition 6.1 *The problems (*0/1-Upgrade-Total-Cost*,* Total Cost*,* Spanning Tree*) and (*0/1-Upgrade-Total-Cost*,* Diameter*,* Spanning Tree*) are* NP*-hard, even if the underlying network G is a tree. This result remains true, even if $c_e(t) = t$ and $\ell^{\min}(e) = 0$ for all $e \in E$.*

Proof: The proof is by a reduction from the Partition problem which is known to be NP-complete [GJ79]. An instance of Partition consists of a set $A = \{x_1, x_2, \ldots, x_n\}$ of integers, where $\sum_{i=1}^{n} x_i$ is even, and the question is whether there is a subset A' of A such that the sum of the integers in A' is equal to $\frac{1}{2}\sum_{i=1}^{n} x_i$. Starting from an instance of Partition, we produce an instance of (0/1-Upgrade-Total-Cost, Diameter, Spanning Tree) (which is also an instance of (0/1-Upgrade-Total-Cost, Diameter, Spanning Tree)) as follows. The graph G is a simple path on $n+1$ nodes. Let v_0, v_1, ..., v_n denote the nodes in the order in which they appear in the path. For edge $e = (v_{i-1}, v_i)$ $(1 \leq i \leq n)$, let $\ell(e) = x_i$, $\ell^{min}(e) = 0$ and $c_e(t) = t$. Further, let the cost budget $B = \frac{1}{2}\sum_{i=1}^{n} x_i$. Since we are considering 0/1-reductions, the cost of upgrading edge e is either 0 or x_i, and the length of e either remains as x_i or is decreased to 0. Using this fact, it can be verified that there is a feasible 0/1-reduction that produces a spanning tree of total length (which is equal to its diameter) $\frac{1}{2}\sum_{i=1}^{n} x_i$ if and only if the Partition instance has a solution. □

Next, we will prove that the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem is NP-hard even for very restricted classes of graphs and the most simple reduction cost functions.

Theorem 6.2 *(*R-Upgrade-Total-Cost*,* Total Cost*,* Spanning Tree*) is* NP*-hard, even when restricted to series-parallel graphs with linear reduction cost functions c_e (i.e., $c_e(t) = c_e \cdot t$ for all $e \in E$).*

Proof: We use a reduction from CONTINUOUS MULTIPLE CHOICE KNAPSACK which is known to be NP-complete (c.f. [GJ79, Problem MP11, page 247]). An instance of CMC-KNAPSACK is given by a finite set U of n items, a size $s(u)$ and value $v(u)$ for each item, a partition $U_1 \cup \cdots \cup U_k$ of U into disjoint sets and two integers S and K. The question is, whether there is a choice of a unique element $u_i \in U_i$, for each $1 \leq i \leq k$, and an assignment of rational numbers $r_i, 0 \leq r_i \leq 1$ to these elements such that $\sum_{i=1}^{k} r_i s(u_i) \leq S$ and $\sum_{i=1}^{k} r_i v(u_i) \geq K$.

Given an instance of CMC-KNAPSACK we construct a graph $G = (V, E)$ in the following way: We let $V = U \cup \{X, T, T_1, \ldots, T_k\}$, $E := E_1 \cup E_2 \cup E_3$ with $E_1 := \{(X, u) : u \in U\}$, $E_2 := \{(u, T_i) : u \in U_i, i = 1, \ldots, k\}$ and $E_3 := \{(T_i, T) : i = 1, \ldots, k\}$. The graph constructed this way is obviously series-parallel with terminals X and T.

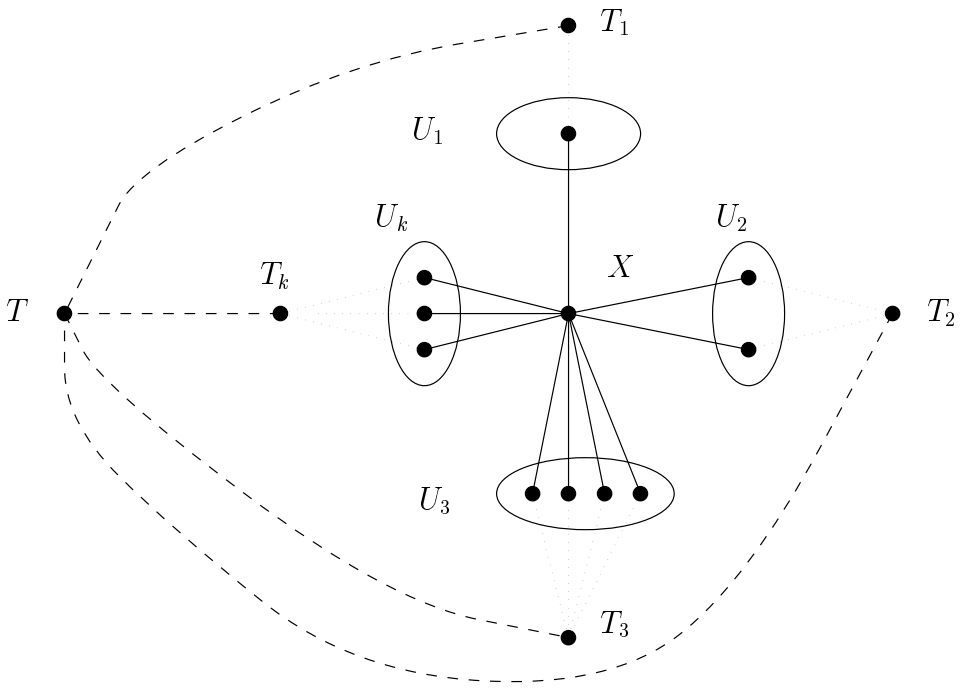


Figure 3: Graph used in the reduction from CONTINUOUS MULTIPLE CHOICE KNAPSACK.

Define $D := \max\{v(u) : u \in U\}$. For each edge $(x, u) \in E_1$, let $\ell(x, u) := D, \ell^{\min}(x, u) := D - v(u), c(x, u) := s(u)/v(u)$. For all edges $e \in E_2$ we let $\ell(e) := \ell^{\min}(e) := c_e := 0$, and for all edges $e \in E_3$ we define $\ell(e) := \ell^{\min}(e) := 3D$ and $c_e := 0$. Set the bound B on the total cost to be S.

The graph is shown in Figure 3. The dotted edges are of weight 0 while the dashed ones have weight $3D$. Any MST in G has weight $kD + 3D$.

By the construction, any feasible reduction can only reduce the length of the edges in E_1. Assume that r is a feasible reduction. Observe that the MST in G with edge lengths given by $(\ell - r)$ will always include *all* edges from E_2 (which are of

weight 0) and *exactly one* edge from E_3, regardless of which edges from E_1 are affected by the reduction. Observe also that for any fixed $i \in \{1, \ldots, k\}$, any MST in the modified graph will contain *exactly one* of the edges of the form (X, u'), where $u' \in U_i$. Consequently, reducing the length of *more than one* edge (X, u') with $u' \in U_i$ will *not* improve the quality of the solution, but cost money from the budget B. We thus have:

Observation: If r is a feasible reduction for the instance of (R-UPGRADE-TOTAL-COST, TOTAL COST, SPANNING TREE) defined above and the weight of an MST in the modified graph is Y, then there is always a feasible reduction r', which for each $i \in \{1, \ldots, k\}$ reduces at most one of the edges (X, u), $u \in U_i$ and the weight of an MST with respect to $(\ell - r')$ is also equal to Y. □

Let r be any reduction as defined in the above observation and for $i = 1, \ldots, k$ let $e_i = (X, u_i)$ be the unique edge from x to U_i affected by the reduction. The weight of an MST in with respect to $(\ell - r)$ is then given by

$$3D + \sum_{i=1}^{k} (\ell(e_i) - r(e_i)) = 3D + k \cdot D - \sum_{i=1}^{k} r(e_i). \tag{2}$$

The cost of reduction r is given by

$$\sum_{i=1}^{k} r(e_i) c_{e_i} = \sum_{i=1}^{k} r(e_i) \cdot \frac{s(u_i)}{v(u_i)} = \sum_{i=1}^{k} \frac{r(e_i)}{v(u_i)} \cdot s(u_i) \leq B. \tag{3}$$

We now prove the following: There is a feasible reduction r such that $\mathrm{MST}_G(\ell - r) \leq (3+k)D - K$, if and only if there exists a choice of a unique element $u_i \in U_i$, $1 \leq i \leq k$ and an assignment of rational numbers $r_i, 0 \leq r_i \leq 1$ to these elements such that $\sum_{i=1}^{k} r_i s(u_i) \leq B$ and $\sum_{i=1}^{k} r_i v(u_i) \geq K$.

First, assume that there is a feasible reduction r such that $\mathrm{MST}_G(\ell - r) \leq (3 + k)D - K$. Without loss of generality, we can assume that r has the properties as stated in the above observation. Then for $i = 1, \ldots, k$ there is at most one edge $e_i = (X, u)$ with $u \in U_i$ such that $r(e_i) > 0$. If there is such an edge e_i, we define

$$r_i := \frac{r(e_i)}{v(u_i)} = \frac{r(e_i)}{\ell(e_i) - \ell^{\min}(e_i)} \tag{4}$$

and let $u_i := u$. If for all edges (X, u) with $u \in U_i$ we have $r(e_i) = 0$, we simply let $r_i := 0$ and choose $u_i \in U$ arbitrarily. It follows readily from the definition and the feasibility of the reduction r that $r_i \in [0, 1]$. Moreover, using Equation (3) we see that $\sum_{i=1}^{n} r_i s(u_i) \leq B \leq B_1$. Using equation (2) and the fact that the weight $\mathrm{MST}_G(\ell - r)$ is no more than $(3 + k)D - K$ we obtain that

$$\sum_{i=1}^{n} r_i v(u_i) = \sum_{i=1}^{n} \frac{r(e_i)}{v(u_i)} \cdot v(u_i) = \sum_{i=1}^{n} r(e_i) \geq K.$$

Conversely, if we can pick unique elements u_i from the sets U_i and find rational numbers $r_i \in [0, 1]$ such that $\sum_{i=1}^{n} r_i s(u_i) \leq B$ and $\sum_{i=1}^{n} r_i v(u_i) \geq K$. We can define a reduction r by $r(X, u_i) := r_i v(u_i) = r_i(\ell(x, u_i) - \ell^{\min}(x, u_i))$ for $i = 1, \ldots, k$ and $r(e) := 0$ for all other edges. It follows that r is indeed feasible, and using equation (2) we see that the MST in the modified graph is no heavier than $(3 + k)D - K$. □

6.2 Non-approximability Results for General Graphs

The above hardness results show that for special classes of graphs, the problems are weakly NP-hard. We now show non-approximability results for several problems for general graphs. These results are obtained by suitable reductions from SET COVER defined below.

Definition 6.3 *An instance of* SET COVER *consists of a set Q of ground elements $\{q_1, \ldots, q_n\}$, a collection $Q_1, \ldots, Q_m$ of subsets of Q and an integer k. The question is whether one can pick at most k sets whose union is equal to Q.*

6.2.1 Results for Total Cost Problems

Theorem 6.4 *Unless* P = NP, *for any $\rho > 1$, there is no polynomial time $(\rho, 1)$ approximation for the (*0/1-UPGRADE-TOTAL-COST, TOTAL COST, STEINER TREE*) problem even when restricted to bipartite graphs.*

Proof: Suppose there is a polynomial time $(\rho, 1)$ approximation algorithm A for the problem for some $\rho > 1$. We will show that A can be used to solve an arbitrary instance of SET COVER.

Given an instance of SET COVER, we construct the natural bipartite graph with one partition for set nodes (denoted by Q_1, Q_2, ..., Q_m) and the other for element nodes (denoted by q_1, q_2, ..., q_n), and edges representing element inclusion in the sets. To this bipartite graph, we add an "enforcer" node (denoted by x) which is adjacent to each of the set nodes. Let G denote the resulting bipartite graph. The set R of terminals for the Steiner tree instance is given by $R = \{x, q_1, q_2, \ldots, q_n\}$. For each edge e in G, we set $\ell(e) = 1$ and $\ell^{\min}(e) = \varepsilon$, where ε is a positive quantity chosen so that

$$\varepsilon < \frac{1}{(\rho - 1)(n + k)}. \tag{5}$$

For each edge, the reduction cost function c_e is given by $c_e(0) = 0$ and $c_e(1 - \varepsilon) = 1$. (Since we are dealing with 0/1-reductions, the cost function needs to be specified only for these two values.) The cost budget B is set to $n + k$, where k is the bound on the size of the set cover.

Suppose there is a set cover $Q' = \{Q_{i_1}, Q_{i_2}, \ldots, Q_{i_k}\}$ of size k. Consider the Steiner tree T in G consisting of x, the edges (x, Q_{i_j}), $1 \leq j \leq k$, and one edge from each element node to some set node in Q'. (Since Q' is a cover, each element node must be adjacent to some set node in Q'.) Let r be the reduction defined by $r(e) = 1 - \varepsilon$ if $e \in T$ and $r(e) = 0$ otherwise. It is easy to see that the total cost of r is $n + k$ and that the total length of T in the modified graph is $(n + k)\varepsilon$. Thus, if there is a set cover of size k, then the modified graph has a Steiner tree of length at most $(n + k)\varepsilon$. Since A is a $(\rho, 1)$ approximation algorithm, the length of a Steiner tree returned by A is at most $\rho(n + k)\varepsilon$.

Suppose there is no set cover of size at most k. Thus, at least $k + 1$ sets are needed to cover the elements. By our construction, any Steiner tree T' that connects together

the $n+1$ nodes in R must have a total of at least $n+k+2$ nodes, and consequently at least $n+k+1$ edges. Since the cost budget is at most $n+k$, there must be at least one edge of length 1 in T' and so the total length of T' is at least $1+(n+k)\varepsilon$.

Using Equation (5), it can be verified that $\rho(n+k)\varepsilon < 1+(n+k)\varepsilon$. Therefore, using A, we can solve an arbitrary instance of Set Cover in polynomial time, contradicting the assumption that P $\neq$ NP. □

Corollary 6.5 *Unless* P = NP*, for any* $\rho > 1$*, there is no polynomial time* $(\rho, 1)$ *approximation for the (*R-Upgrade-Total-Cost*,* Total Cost*,* Steiner Tree*) and (*I-Upgrade-Total-Cost*,* Total Cost*,* Steiner Tree*) problems even when restricted to bipartite graphs.*

Proof: Let us first consider the (R-Upgrade-Total-Cost, Total Cost, Steiner Tree) problem. We use the same construction as above, except that for each edge e the reduction cost function c_e is given by $c_e(t) = t/(1-\varepsilon)$ for $t \geq 0$, where ε satisfies Equation (5). By the same argument as in the proof of Theorem 6.4, it follows that if there is a set cover of size k, a $(\rho, 1)$-approximation algorithm must return a reduction r and a Steiner tree of length at most $\rho(n+k)\varepsilon$ under the modified edge lengths.

Conversely, if there is no set cover of size at most k, then again any Steiner tree T in the graph must contain at least $n+k+1$ edges. Since for t units of the budget the weight of T can be reduced by at most $(1-\varepsilon)t$ units, it follows that after modifying the lengths for a budget of $n+k$ the weight of T is at least $1+(n+k)\varepsilon$. The remainder of the proof is identical to that in the proof of Theorem 6.4.

For (I-Upgrade-Total-Cost, Total Cost, Steiner Tree), we let $\ell^{\min}(e) = 1$ and $\ell(e) = 1+(\rho-1)(n+k)$ for all edges in the graph. We also define $c_e(t) = t$ and set the budget to $(\rho-1)(n+k)^2$. Now, the remainder of the proof is along the same lines as above. □

The following complementary non-approximability result for the above problems is a direct consequence of the fact that the optimal Steiner tree problem is NP-hard even for bipartite graphs [GJ79, Problem ND12, pages 208–209].

Observation 6.6 *Unless* P = NP*, for any* $\rho > 1$*, there is no polynomial time* $(1, \rho)$ *approximation algorithm for the (*0/1-Upgrade-Total-Cost*,* Total Cost*,* Steiner Tree*), (*I-Upgrade-Total-Cost*,* Total Cost*,* Steiner Tree*) and (*R-Upgrade-Total-Cost*,* Total Cost*,* Steiner Tree*) problems even when restricted to bipartite graphs.* □

6.2.2 Results for Diameter Problems

Our next proposition presents a negative result concerning the approximability of (I-Upgrade-Total-Cost, Diameter, Spanning Tree) and (0/1-Upgrade-Total-Cost, Diameter, Spanning Tree) problems. These results are obtained using a reduction from the Set Cover problem and the following hardness result from [Fe95] for Min Set Cover, an optimization version of Set Cover.

Theorem 6.7 *Unless* NP $\subseteq$ DTIME($N^{\log\log N}$), *for any* $0 < \mu < 1$, *the* Min Set Cover *problem, with a universe of size* K, *cannot be approximated in polynomial time to within a* $\mu \ln K$ *factor.* □

Proposition 6.8 *Unless* NP $\subseteq$ DTIME($N^{\log\log N}$), *for any* $\varepsilon > 0$ *and* $0 < \mu < 1$, *there is no polynomial time* $(11/10 - \varepsilon, \mu \ln B)$ *approximation algorithm for either of the problems (*0/1-Upgrade-Total-Cost, Diameter, Spanning Tree*) and (*I-Upgrade-Total-Cost, Diameter, Spanning Tree*).*

Proof: Given an instance of Set Cover, we first construct the natural bipartite graph, with one side of the partition for set nodes Q_j, $j = 1, \ldots, m$, and the other for element nodes q_i, $i = 1, \ldots, n$. We insert an edge $\{Q_j, q_i\}$ iff $q_i \in Q_j$. All these edges e have length $\ell(e) = \ell^{min}(e) = 4$. Now we add an enforcer node x and join it to all the set nodes. For these edges e we define $\ell(e) = 2$, $\ell^{min}(e) = 1$. Finally, for each edge e, we let $c_e(0) = 0$ and $c_e(t) = 1$ for any $t > 0$, and choose $B = k$.

The above construction yields both an instance of (I-Upgrade-Total-Cost, Diameter, Spanning Tree) and an instance of (0/1-Upgrade-Total-Cost, Diameter, Spanning Tree). Without loss of generality, we can assume that there is no single set Q_j covering all the elements in Q; i.e., $Q_j \neq Q$ for $j = 1, \ldots, m$. Then the spanning tree T^* in G' with minimum diameter satisfies $\mathrm{dia}(T^*) = 12$, and a diametric path of that tree is between any two element nodes that are not adjacent to the same set node.

Observe that any feasible reduction r corresponds to a choice of at most $B = k$ sets from the collection $Q_1, \ldots, Q_m$.

Given any integer reduction r it is easy to see that there is a spanning tree in G' (with edge lengths given by $\ell - r$) with diameter 10, if the selection of sets corresponding to the reduction covers all the elements in Q, and that the diameter of G' (again with edge lengths given by $\ell - r$) is at least 11, if the selection does not form a cover.

For expository reasons, we first argue that, unless P = NP, there is no polynomial time $(11/10 - \varepsilon, 1)$ approximation algorithm for the problem. Suppose A is such an algorithm. If there is a set cover of size k or less, then A must return a reduction A(r) yielding a spanning tree of diameter at most 10. On the other hand, if there is no set cover, then the best tree we can obtain by modifying the network has diameter 11. Thus Algorithm A can be used to decide an arbitrary instance of Set Cover.

We now turn to the proof of the result stated in the proposition. Suppose A is an algorithm that provides a performance guarantee of $(11/10 - \varepsilon, \mu \ln B)$ for some $\varepsilon > 0$ and $0 < \mu < 1$. Given any instance I of Min Set Cover, we construct the graph G' as above. Then we run the algorithm A for the budgets $B = 1, \ldots, \min\{n, m\}$. Observe that this will still result in an overall polynomial time. Let $B_{\min}$ denote the minimum budget in $\{1, \ldots, \min\{n, m\}\}$ such that A returns a reduction A(r) resulting in a spanning tree of diameter 10. By the above observations and the fact that the algorithm spends at most $\mu B \ln B$ units of money, we see that there must be a set cover of size at most $\mu B \ln B$. By the choice of $B_{\min}$ there can be no set cover of size

strictly less than $B_{\min}$. Thus we can approximate the minimum set cover by a factor of no more than $\mu \ln B_{\min} \leq \mu \ln n$. □

7 Approximation Algorithm for General Graphs

In this section, we present our approximation algorithm for the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem. As mentioned earlier, the approximation algorithm extends easily to a broad class of network improvement problems where the objective to be minimized is the total cost of a connected subnetwork (e.g. budget constrained minimum Steiner tree problem).

7.1 High Level Description

We first give an informal description of the algorithm. The main procedure uses a parametric search. In this search, the algorithm tries to find a good compromise between weighing the total length and the corresponding reduction cost of a tree in general. To this end, the algorithm performs a binary search with parameter K on the interval $\mathcal{I} := [\frac{1}{\gamma}(n-1)\min_{e\in E} \ell_{\min}(e), \frac{1}{\gamma}(n-1)\max_{e\in E} \ell(e)]$. Note that if $\mathrm{MST}_G(\ell - r^*)$ denotes the total weight of a minimum spanning tree after an optimal reduction r^* then $\frac{1}{\gamma}\mathrm{MST}_G(\ell - r^*) \in \mathcal{I}$.

For each $K \in \mathcal{I}$, which is probed with the help of a test procedure during the search, the algorithm first calculates a coarse heuristic measure that indicates how important it is to shorten an edge. Then, for each edge e in the graph, the blend of its length and the reduction cost is refined using the cost function c_e. After calculating such *compound costs* for the edges, we compute a minimum spanning tree with respect to these costs. The algorithm stops when a good blend has been found, meaning in this context that there exists a tree of total compound cost that is small compared to the current parameter K.

For large values of K the reduction costs on the edges are weighted more than their lengths and the algorithm will tend to reduce the edge lengths only by a small amount, resulting in low overall reduction costs and more or less heavy trees. Also, since K is large, the test on the compound cost of the minimum spanning tree computed will succeed. The algorithm now tries to reduce K as much as possible and find a minimum $K \in \mathcal{I}$ such that it can successfully compute a light compound cost spanning tree.

Our approximation algorithm for (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) is shown in Figure 4. This algorithm uses the test procedure given in Figure 5.

7.2 Correctness and Performance Guarantee

We now turn to prove the performance guarantee provided by the algorithm Heuristic-Upgrade. We first prove some preliminary lemmas.

HEURISTIC-UPGRADE(γ, ε)

1 Perform a binary search on the interval

$$\mathcal{I} = \left[\frac{(n-1) \min\limits_{e \in E} \ell_{\min}(e)}{\gamma}, \frac{(n-1) \max\limits_{e \in E} \ell(e)}{\gamma} \right] \tag{6}$$

with a spacing of ε to find the minimum value $K' \in \mathcal{I}$ such that TEST-BLEND(K') returns "Yes".

2 Let T' be the tree generated by TEST-BLEND(K') and let t_e ($e \in T'$) be the corresponding "fine tuned" blend parameters.

3 Define the reduction r by $r(e) := 0$ if e is not included in T and by $r(e) := t_e$ otherwise.

4 **return** r and T.

Figure 4: Main Procedure for the approximation of (R-UPGRADE-TOTAL-COST, TOTAL COST, SPANNING TREE).

TEST-BLEND(K)

1 **Comment:** This procedure tries to estimate whether in the current blend of lengths and reduction costs, the costs are weighted strongly enough (i.e., K large enough) resulting in a low cost reduction. For this purpose, it uses the heuristic measure computed in Step 2.

2 **for** each edge e let $h_K(e) = \min\limits_{t \in [0, \ell(e) - \ell^{\min}(e)]} \left(\ell(e) - t + \frac{K}{B} c_e(t) \right)$.

Also, let t_e be the value of t which achieves the value $h_K(e)$.

3 Compute a minimum spanning tree T in G using the weight $h_K(e)$ for each $e \in E$. Let $h_K(T)$ denote the cost of this spanning tree.

4 **if** $h_K(T) \leq (1+\gamma)K$ **then return** "Yes" **else return** "No".

Figure 5: Test procedure used for the approximation of (R-UPGRADE-TOTAL-COST, TOTAL COST, SPANNING TREE).

The proof of performance relies mainly on the following lemma, which ensures that the binary search in the main procedure works correctly. In stating this lemma, we use the notation introduced in the two procedures (HEURISTIC-UPGRADE and TEST-BLEND) described above.

Lemma 7.1 *Define F on $\mathbb{R}_{>0}$ by $F(K) := \frac{\mathrm{MST}_G(h_K)}{K}$. Then F is monotonically non-increasing on $\mathbb{R}_{>0}$.*

Proof: Let $K^{(1)}$ and $K^{(2)}$ be two positive numbers such that $K^{(1)} < K^{(2)}$. For $i = 1, 2$ let $T^{(i)}$ be a minimum spanning tree in G under the cost function $h_{K^{(i)}}$. Then

$$h_{K^{(i)}}(T^{(i)}) = \sum_{e \in T^{(i)}} h_{K^{(i)}}(e) = \underbrace{\sum_{e \in T^{(i)}} (\ell(e) - t_e^{(i)})}_{=:\ L^{(i)}} + \frac{K^{(i)}}{B} \underbrace{\sum_{e \in T^{(i)}} c_e(t_e^{(i)})}_{=:\ C^{(i)}} =: \ell^{(i)} + \frac{K^{(i)}}{B} C^{(i)}. \tag{7}$$

Here $t_e^{(i)}$ are the values chosen in Step 2 of TEST-BLEND which minimize $\ell(e) - t + \frac{K^{(i)}}{B} c_e(t)$ on the interval $[0, \ell(e) - \ell^{\min}(e)]$. By dividing the last equation by $K^{(i)}$ we obtain that

$$F(K^{(i)}) = \frac{L^{(i)}}{K^{(i)}} + \frac{C^{(i)}}{B} \qquad \text{for } i \in \{1, 2\}. \tag{8}$$

In the next step we find an upper bound for $F(K^{(2)})$. To this end, we estimate the weight of each edge in $T^{(1)}$ under the cost function $h_{K^{(2)}}$. Let $e \in T^{(1)}$ be an arbitrary edge. Then by the choice of $t_e^{(2)}$ in Step 2 of TEST-BLEND we have that

$$\begin{aligned} h_{K^{(2)}}(e) &= \ell(e) - t_e^{(2)} + \frac{K^{(2)}}{B} c_e(t_e^{(2)}) \\ &= \min_{t \in [0, \ell(e) - \ell^{\min}(e)]} \left(\ell(e) - t + \frac{K^{(2)}}{B} c_e(t) \right) \\ &\leq \ell(e) - t_e^{(1)} + \frac{K^{(2)}}{B} c_e(t_e^{(1)}). \end{aligned} \tag{9}$$

Summing up the inequalities in (9) over all $e \in T^{(1)}$, we obtain:

$$h_{K^{(2)}}(T^{(1)}) \leq L^{(1)} + \frac{K^{(2)}}{B} C^{(1)}. \tag{10}$$

Dividing (10) by $K^{(2)}$ and using the fact that $h_{K^{(2)}}(T^{(2)}) \leq h_{K^{(2)}}(T^{(1)})$ this results in

$$F(K^{(2)}) \leq \frac{L^{(1)}}{K^{(2)}} + \frac{C^{(1)}}{B} < \frac{L^{(1)}}{K^{(1)}} + \frac{C^{(1)}}{B} = F(K^{(1)}). \tag{11}$$

The strict inequality in the chain above stems from the fact that $K^{(1)} < K^{(2)}$. This completes the proof of the lemma. □

Corollary 7.2 *If the procedure* Test-Blend *returns "*Yes*" for some $K' > 0$ then it also returns "*Yes*" for all $K > K'$. Thus, the binary search in* Heuristic-(R-Upgrade-Total-Cost, Total Cost, Spanning Tree) *works correctly.*

Proof: Let T' be a minimum spanning tree with respect to $h_{K'}$. Then, since the test procedure Test-Blend(K') returns "Yes" we have that $h_{K'}(T') \leq (1+\gamma)K'$; i.e., $F(K') \leq (1+\gamma)$. Thus it follows by Lemma 7.1 that $F(K) \leq (1+\gamma)$ for all $K > K'$. Since $F(K) = \frac{\mathrm{MST}_G(h_K)}{K}$, this is equivalent to saying that $\mathrm{MST}_G(h_K) \leq (1+\gamma)K$ for all $K > K'$. □

We now prove the performance of the algorithm.

Theorem 7.3 *For any fixed $\gamma, \varepsilon > 0$,* Heuristic-Upgrade *is an approximation algorithm for (*R-Upgrade-Total-Cost, Total Cost, Spanning Tree*) that finds a solution whose length is at most $(1+\frac{1}{\gamma})$ times that of a minimum length spanning tree plus an additive constant of at most ε, and the total cost of the improvement is at most $(1+\gamma)$ times the budget B.*

Proof: Let r^* be an optimal feasible reduction and let T^* be a minimum spanning tree in G with respect to the weight function $\ell - r^*$. For the sake of shorter notation let $L^* := (\ell - r^*)(T^*)$ be its total weight in the graph with the edge lengths resulting from the optimal reduction r^*.

We now show Test-Blend would return "Yes" if called with the value $\tilde{K}$ which is the smallest value in the ε-spacing of the interval $\mathcal{I}$ from (6) satisfying $\tilde{K} \geq L^*/\gamma$. Thus, $\tilde{K}$ is some rational number satisfying the equation

$$\tilde{K} = L^*/\gamma + \varepsilon' \tag{12}$$

where $0 \leq \varepsilon' < \varepsilon$.

For each edge $e \in T^*$ we can estimate the weight $h_{K^*}(e)$ similar to inequality (9) in the proof of Lemma 7.1. This way, we see that the weight of T^* under $h_{\tilde{K}}$ is no more than $L^* + \frac{\tilde{K}}{B}B$. Consequently, the minimum spanning tree with respect to $h_{\tilde{K}}$ that would be found by the procedure during the call has $h_{\tilde{K}}$-weight at most

$$L^* + \tilde{K} \overset{(12)}{=} \gamma(\tilde{K} - \varepsilon') + \tilde{K} \leq (1+\gamma)\tilde{K}.$$

Hence, the test in Step 4 of Test-Blend would be successful and the procedure would return "Yes". Since we know by Corollary 7.2 that the binary search correctly locates a minimum value K', this now implies that the minimum value K' must satisfy $K' \leq \tilde{K} = L^*/\gamma + \varepsilon'$. Let T' be the minimum spanning tree found by Test-Blend(K'). Since $K', B \geq 0$ and $c_e(t) \geq 0$ for all t, we have:

$$h_{K'}(T') = \sum_{e \in T'} (\ell(e) - t'_e) + \frac{K'}{B} \sum_{e \in T'} c_e(t'_e) \geq \sum_{e \in T'} (\ell(e) - t'_e). \tag{13}$$

Here again the numbers t'_e are the values of t chosen in Step 2 of the test procedure. For the reduction r which is calculated in Step 3 of HEURISTIC-(R-UPGRADE-TOTAL-COST, TOTAL COST, SPANNING TREE) it now follows from (13) that

$$\mathrm{MST}_G(\ell - r) \leq (\ell - r)(T') \leq h_{K'}(T'). \tag{14}$$

Moreover,

$$\begin{aligned} h_{K'}(T_{K'}) &\leq h_{K'}(T^*) \leq L^* + \frac{K'}{B}B \leq L^* + \tilde{K} \\ &\overset{(12)}{=} L^* + \frac{L^*}{\gamma} + \varepsilon' \leq (1 + \frac{1}{\gamma})L^* + \varepsilon \\ &= (1 + \frac{1}{\gamma})\mathrm{MST}_G(\ell - r^*) + \varepsilon. \end{aligned}$$

Using this result in (14), we get $\mathrm{MST}_G(\ell - r) \leq (1 + \frac{1}{\gamma})\mathrm{MST}_G(\ell - r^*) + \varepsilon$, which proves the claimed performance of the algorithm with respect to the weight of an MST in the graph after applying the reduction r.

We now estimate the cost of the reduction r found by our heuristic. Note that the cost of r is exactly $\sum_{e \in T'} c_e(t_e)$. We have

$$\frac{K'}{B} \sum_{e \in T'} c_e(t'_e) \leq \sum_{e \in T'} (\ell(e) - t'_e + \frac{K'}{B} c_e(t'_e)) = h_{K'}(T') \leq (1 + \gamma)K'.$$

Dividing the last chain of inequalities by $\frac{K'}{B}$ yields that the budget B is violated by a factor of at most $(1 + \gamma)$ as claimed in the theorem. □

7.3 Running Time

We now show that the algorithm can be implemented to run in polynomial time for a broad class of reduction cost functions c_e on the edges of the graph. Let $L_{\max} = \max_{e \in E} \ell(e)$. Then the total number of calls to Procedure TEST-BLEND is $\mathcal{O}(\log(\frac{nL_{\max}}{\gamma\varepsilon}))$. Since γ and ε are fixed, the test procedure is called only a polynomial number of times. Thus, to prove that the overall running time of the algorithm is polynomial, it suffices to show that each execution of TEST-BLEND can be completed in polynomial time. Here, the only condition to show is that we can minimize the function $f_e(t) := \ell(e) - t + \frac{K}{B} c_e(t)$ on the compact interval $\mathcal{I}' := [0, \ell(e) - \ell^{\min}(e)]$ in Step 2 of the procedure in polynomial time. The rest of the procedure consists of computing a minimum spanning tree which can be done in $\mathcal{O}(n + m \log \beta(m, n))$ time using the algorithm of Gabow et. al. [GGST86], where $\beta(m, n) = \min\{i \mid \log^{(i)} n \leq m/n\}$.

Consider the execution of TEST-BLEND for a given value of K. Observe that in Step 2 the number $\ell(e)$ is an additive constant and $\frac{K}{B}$ is a constant factor. Thus, the constrained minimization of f_e can be done easily for for the following sample classes of functions c_e:

1. Linear functions, that is, $c_e(t) = c_e \cdot t$ for some constant c_e: Then f_e is a linear function in t and the minimum is attained at one of the endpoints of $\mathcal{I}'$. Minimizing f_e can be done in constant time. Thus, the total running time of the heuristic is $\mathcal{O}(\log(\frac{nL_{\max}}{\gamma\varepsilon})(n + m \log \beta(m, n)))$. We will show in Section 8 how to improve the algorithm for this particular class of functions.

2. Concave functions: Let $\Delta(e) := \ell(e) - \ell^{\min}(e)$. Then, for any $0 < \lambda < 1$ we have by the concavity of c_e (which implies the concavity of f_e):

$$f_e(\lambda \cdot 0 + (1 - \lambda)\Delta(e)) \geq \lambda f_e(0) + (1 - \lambda) f_e(\Delta(e)) \geq \min \{f_e(0), f_e(\Delta(e))\} .$$

   Thus, the minimum of f_e is again either at 0 or at $\ell(e) - \ell^{\min}(e)$.

3. Differentiable convex functions where we can find a root of the equation $c'_e(t) = \frac{B}{K}$ explicitly.

4. Functions that are piecewise of one of the types described above. Observe that the number of pieces is polynomial in the input size.

For the first three classes of functions mentioned above, the total computational effort of our algorithm consists essentially of $\mathcal{O}(\log(\frac{nL_{\max}}{\gamma\varepsilon}))$ minimum spanning tree computations, which results not only in an overall polynomial time but also in a complexity that is feasible in practice.

7.4 Notes on the Algorithm

It should be noted that our Algorithm Heuristic-Upgrade can be modified easily to handle instances of (I-Upgrade-Total-Cost, Total Cost, Spanning Tree) and (0/1-Upgrade-Total-Cost, Total Cost, Spanning Tree), that is, the cases where the reduction is required to be either integer valued or to satisfy $r(e) \in \{0, \ell(e) - \ell^{\min}(e)\}$ for all $e \in E$. For these cases, Step 2 of Test-Blend is modified in such a way that the minimization is carried out only over the integers in $[0, \ell(e) - \ell^{\min}(e)]$ or on the two element set $\{0, \ell(e) - \ell^{\min}(e)\}$ respectively.

Integer valued reductions are helpful to model discrete steps of improvement, e.g. the addition of a number of communication links parallel to existing ones in the network. 0/1-Reductions can be used the model the insertion of alternative edges to the graph G, with the reduction of the edge e corresponding to the construction of a new edge e' parallel to e with length $\ell^{\min}(e)$.

So far, we have assumed that the function $f_e(t) = \ell(e) - t + \frac{K}{B} c_e(t)$ can be minimized *exactly*. This indeed is not necessary to obtain a constant factor approximation algorithm for (R-Upgrade-Total-Cost, Total Cost, Spanning Tree). In fact, one can show that if in Step 2 of procedure Test-Blend we find a value t' satisfying

$$f_e(t') \leq \alpha \cdot \min_{t \in [0, \ell(e) - \ell^{\min}(e)]} f_e(t)$$

for some $\alpha \geq 1$ and modify Step 4 to check whether the compound weight of the tree is at most $\alpha^2(1+\gamma)K$, this will lead to a polynomial time algorithm which produces a reduction of cost at most $\alpha^2(1+\gamma)B$ and a corresponding MST of total length at most $\alpha(1+1/\gamma)$ times that of an optimal tree plus an additive constant of ε.

8 Faster Algorithm for Linear Reduction Costs

In this section we show how to improve the performance and the running time of the approximation algorithm from Section 7 in the case that the reduction costs on the edges are *linear*; i.e., $c_e(t) \equiv c_e \cdot t$ for all $e \in E$.

The first observation for the improved algorithm is the following: In Step 2 of procedure Test-Blend the linear function $f_e(t) := \ell(e) + t(\frac{K}{B}c_e - 1)$ is minimized over the interval $[0, \Delta(e)]$, where again $\Delta(e) = \ell(e) - \ell^{\min}(e)$. At which of the two endpoints of the interval the minimum is attained depends solely on the factor $\frac{K}{B}c_e - 1$. If $\frac{K}{B}c_e - 1 \leq 0$, that is, if $K \leq B/c_e$, then f_e attains its minimum at $\Delta(e)$. Otherwise, f_e is minimized at 0.

8.1 The Structure of the Compound Weights

Observe that the h_K-weight of an edge e is given by

$$h_K(e) := \begin{cases} \ell^{\min}(e) + K\frac{(\ell(e)-\ell^{\min}(e))c_e}{B} & \text{if } K < B/c_e \\ \ell(e) & \text{if } K \geq B/c_e. \end{cases} \tag{15}$$

Thus, for each edge e, the compound weight $h_K(e)$ viewed as a function of K is a linear function with exactly one breakpoint at B/c_e. For $K \geq B/c_e$, the function has the constant value $\ell(e)$, while for $K \leq B/c_e$ it has slope $\frac{(\ell(e)-\ell^{\min}(e))c_e}{B}$.

If we plot the compound weight $h_K(e)$ for each edge $e \in E$, for increasing K we get a linear function with exactly one breakpoint. This breakpoint is at B/c_e. Figure 6 shows an example of plots of these compound weights.

It is easy to see that, given two edges e and e', their ordering with respect to the compound weights h_K changes at most twice when K varies. Also, these at most two values of K, can be computed in constant time.

8.2 The Basic Idea for the Improved Algorithm

Let $K^* \in \mathcal{I}$ be the overall minimum value such that Test-Blend(K) would return "Yes" if called with $K = K^*$ (the interval $\mathcal{I}$ is defined in (6)). We can use the the analysis from Section 7 to show that $K^* \leq L^*/\gamma$, where L^* again denotes the length of an optimal reduced tree for a budget of B.

We now have the following important lemma:

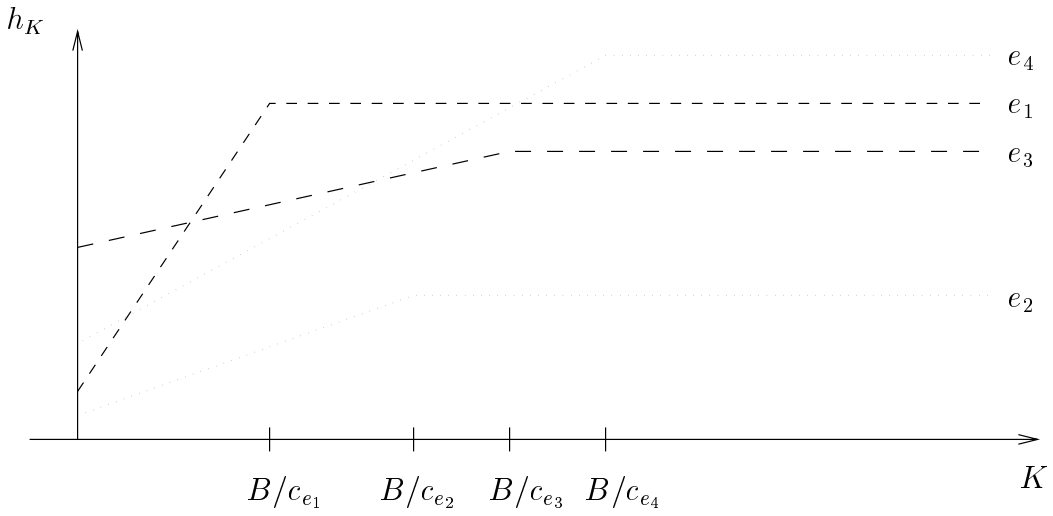


Figure 6: Compound weight h_K of edges for increasing K.

Lemma 8.1 *If the ordering of the edges with respect to their h_{K^*}-weights is known, we can construct a tree T and a reduction r in time $\mathcal{O}(n + m \log \beta(m, n))$ with the following properties:*

(i) The cost $\sum_{e \in E} c_e r(e)$ of the reduction r is at most $(1 + \gamma)B$.

(ii) The weight $(\ell - r)(T)$ in the modified graph is no more than $(1 + 1/\gamma)L^*$.

Proof: Observe that, if we knew the h_{K^*}-weights of the edges in the tree T, we could now construct a reduction r' just as in Step 3 of Heuristic-Upgrade. Using exactly the same argument as in the proof of Theorem 7.3 but now using that $K^* \leq L^*/\gamma$ instead of $K' \leq L^*/\gamma + \varepsilon$, it then follows that

$$(\ell - r')(T) \leq (1 + 1/\gamma)L^*,$$

and that the cost of the reduction r' is at most $(1 + \gamma)B$.

But by the assumption of the lemma, we only have knowledge only about the *ordering* of the edges and *not* about K^* or h_{K^*}. We overcome this problem as follows.

Given the ordering of the edges according to their weights, we can use the minimum spanning tree algorithm of Gabow et. al. [GGST86] to compute a minimum spanning tree with respect to the h_{K^*}-weights, without actually knowing these weights. The ordering suffices for this purpose.

Recall from Section 5 that, given a tree and a budget, we can construct an optimal reduction on the tree for that budget in $\mathcal{O}(n)$ time by a Greedy-type algorithm that repeatedly reduces the length of the cheapest edge until the budget is exhausted. Thus, if we compute such a reduction r on our tree T with the budget set to $(1 + \gamma)B$, the length of T under $\ell - r$ will be at most $(\ell - r')(T)$, which in turn is bounded from above by $(1 + 1/\gamma)L^*$. □

Lemma 8.1 suggests finding an ordering of the edges in the graph according to their compound weight at K^*. In the sequel we will show how using a technique of Megiddo [Meg83] this can be accomplished efficiently.

Basically we wish to sort the set $S := \{h_{K^*}(e_1), \ldots, h_{K^*}(e_m)\}$ where K^* is not known. However, for any K we can decide whether $K^* \leq K$ or $K^* > K$ by one MST computation: We compute an MST with respect to edge weights given by h_K and compare its weight to $(1+\gamma)K$. If the weight is bounded from above by $(1+\gamma)K$, then we know that $K^* \leq K$. Otherwise, we can conclude that $K^* > K$.

To simplify the presentation, we will first sketch the main idea before going into details. Imagine applying a (sequential) sorting algorithm to S. The sorting algorithm would start by comparing some values $h_{K^*}(e)$ and $h_{K^*}(e')$. Then, we could do the the following: We compute the values of K such that the ordering of e and e' with respect to the compound weight h_K changes. As seen earlier these are at most two values of K. We compute a minimum spanning tree for each of these "critical values" $K_{e,e'}$, and then decide whether $K^* \leq K_{e,e'}$ or $K^* > K_{e,e'}$. Since the ordering of the edges e and e' only changes at the intersection points, we can decide whether $h_{K^*}(e) \geq h_{K^*}(e')$ or vice versa. Thus, by $\mathcal{O}(1)$ MST computations we can answer a comparison.

Using the idea from above in conjunction with a standard sequential sorting algorithm (which makes $\mathcal{O}(m \log m)$ comparisons), we could find the ordering of the edges at K^* by $\mathcal{O}(m \log m)$ MST computations. However, using Megiddo's technique from [Meg83] we can speed up the algorithm substantially.

8.3 Finding the Ordering with Respect to h_{K^*} Faster

The crucial trick is to use a clever adaption of a sequentialized parallel sorting algorithm such as Cole's scheme [Col88]. Recall that a comparison essentially consists of a MST computation, so comparisons are expensive. Using the parallel sorting scheme, we basically accept a greater total number of comparisons, but we can use the parallelism to group the independent comparisons made in one stage of the parallel machine and then answer all of them together efficiently.

Cole's algorithm uses m processors to sort an array of m elements in parallel time $\mathcal{O}(\log m)$. Recall that in our case $m = |E|$ is the number of edges in the graph $G = (V, E)$. The algorithm is simulated serially, employing one "processor" at a time, according to some fixed permutation, letting each perform one step in each cycle. When two values $h_{K^*}(e)$ and $h_{K^*}(e')$ have to be compared, we compute the at most two critical values where the ordering changes (but we do *not* answer the comparison yet). The crucial observation is that the critical values can be computed independently, meaning that each of the "processors" does not need any knowledge about the critical points computed by the other ones.

After the first of the $\mathcal{O}(\log m)$ stages, we are given at most $2m$ critical values of K, say $K_1 \leq K_2 \leq \cdots \leq K_r$ with $r \leq 2P$. For convenience set $K_0 := -\infty$ and $K_{r+1} := +\infty$. Using binary search, we find an interval $[K_i, K_{i+1}]$, where K^* must be contained.

This is done in the following way: Start with $low := -\infty$ and $high := +\infty$. Then compute the median $M := K_{\lfloor (r+1)/2 \rfloor}$ of the K_j in $\mathcal{O}(r)$ time. We then decide whether $K^* \leq M$ by computing a MST T with edge weights given by h_M: If $h_M(T) \leq (1+\gamma)M$, then we know that $K^* \leq M$. Otherwise, $K^* > M$. In the first case, we set $high := M$ and remove all values K_j with $K_j > M$ from our set of critical values. Similarly, in the second case we set $low := M$ and remove the values smaller than the median M. Clearly, this can be done in $\mathcal{O}(r)$ time. Since M was the median of the K_j the number of critical values decreases by a factor of one half.

Then, the total time effort $Time(r)$ for the binary search satisfies the recurrence:

$$Time(r) = Time(r/2) + T_{MST} + \mathcal{O}(r),$$

where T_{MST} is the time needed for one MST computation. The solution of the recurrence is $Time(r) = \mathcal{O}(r + T_{MST} \log r)$. Since $r \in \mathcal{O}(m)$, this shows that we obtain the interval $[K_i, K_{i+1}]$ containing K^* by $\mathcal{O}(\log m)$ MST computations plus an overhead of $\mathcal{O}(m)$ elementary operations.

Notice that by construction the interval $[K_i, K_{i+1}]$ does not contain any critical points in the interior. If $K_i = K_{i+1}$, then we know that $K^* = K_i = K_{i+1}$. This way we have determined K^*. In this case we can compute the order of all edges with respect to h_{K^*} in $\mathcal{O}(m \log m)$ time and stop the modified sorting algorithm. Lemma 8.1 then enables us to compute a reduction with the properties (i) and (ii) stated there.

Otherwise, the interior of $[K_i, K_{i+1}]$ is nonempty. We compute a minimum spanning tree T with respect to h_{K_i} and test whether $h_{K_i}(T) \leq (1+\gamma)K_i$. If this is the case, then $K^* \leq K_i$, which implies that $K^* = K_i$ since we know that $K^* \in [K_i, K_{i+1}]$. Again, the adopted sorting procedure can stop after having computed the ordering of all edges with respect to h_{K^*}.

The remaining case is that $K_i < K^* \leq K_{i+1}$. In this case it is easy to see that answering the comparisons from the first round by inspecting the weights h_τ, where $\tau \in (K_i, K_{i+1})$ is any interior point of the interval $[K_i, K_{i+1}]$, gives the same results as answering the comparisons with respect to the h_{K^*}-weights.

Thus, at the end of the the first round, our algorithm has either found K^* and thus the ordering of *all* edges in the graph with respect to their h_{K^*}-weights, or we can answer the comparisons from the first round using the ordering of the edges with respect to h_τ.

The above process is repeated $\mathcal{O}(\log m)$ times, once for each parallel step of the parallel sorting machine. Since in each of the $\mathcal{O}(\log m)$ rounds we answer all comparisons of the parallel sorting scheme, upon termination we have found the ordering of the edges with respect to the h_{K^*}-weights. We then use Lemma 8.1 to compute a reduction strategy r and a tree T.

The time needed for the algorithm above can be estimated as follows: There are $\mathcal{O}(\log m)$ cycles altogether. In each round we evaluate $\mathcal{O}(m)$ intersection points. Also, we need $\mathcal{O}(\log m)$ minimum spanning tree computations plus the overhead of $\mathcal{O}(m)$. This results in an overall time of $\mathcal{O}(m \log m + T_{MST} \log^2 m)$, where $T_{MST} = \mathcal{O}(n +$

$m \log \beta(m,n))$ is the time needed for computing a minimum spanning tree. This gives us the following theorem:

Theorem 8.2 *For any fixed $\gamma > 0$ the algorithm presented above is a $(1+1/\gamma, 1+\gamma)$-approximation algorithm for (*R-Upgrade-Total-Cost, Total Cost, Spanning Tree*) with linear reduction costs. The running time of the algorithm is $\mathcal{O}(n \log^2 n + m \log^2 n \log \beta(n,m))$.* □

9 Extension to Steiner Trees and Other Networks

The technique used to obtain results for the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem in Section 7 is quite general. Specifically, Given *any* ρ-approximation algorithm for finding a subgraph $S \in \mathcal{S}$ minimizing the objective Total Cost, the method allows us to obtain a $((1+1/\gamma)\rho, (1+\gamma)\rho)$-approximation algorithm for the (R-Upgrade-Total-Cost, Total Cost, $\mathcal{S}$) problem. For example, using this technique in conjunction with the results of Goemans et al. [GGP+94], we get the first approximation results for edge based improvement problems such as finding minimum-cost generalized Steiner trees, minimum k-edge connected subgraphs, or other network design problems specified by weakly supermodular functions. Thus for example, we get $(O(1), O(1))$-approximation algorithms for the (R-Upgrade-Total-Cost, Total cost, Generalized Steiner tree) and (R-Upgrade-Total-Cost, Total cost, k-edge connected subgraph) problems. (See [AK+95, GW92, KV+94] for the results on the corresponding unicriterion problems.) We illustrate these extensions by briefly discussing the modifications necessary for obtaining an approximation algorithm for the (R-Upgrade-Total-Cost, Total Cost, Steiner Tree) problem. Let Approx-Steiner denote a ρ approximation algorithm for the Steiner tree problem. (For example, we can use the 11/6 approximation algorithm by Zelikovsky [Ze94].) In Step 3 of the test procedure displayed in Figure 5, we call Approx-Steiner to compute an approximate solution to the Steiner tree problem in G using the weight $h_K(e)$ for each $e \in E$. Let MSTEINERT$_G$ denote a minimum cost Steiner tree in G. Then, it is straightforward to see that Lemma 7.1 holds even if replace MST_G by MSTEINERT$_G$. Finally, it is easy to see that proof of Theorem 7.3 carries over with an additional factor of ρ in the performance for the budget as well as the cost of the tree. Thus, for all $\gamma > 0$, and for some $1 \leq \rho < 2$, we get a $((1+1/\gamma)\rho, (1+\gamma)\rho)$ approximation algorithm for the (R-Upgrade-Total-Cost, Total Cost, Steiner tree) problem. In contrast, recall (Theorem 6.4) that unless $\mathsf{P} = \mathsf{NP}$, for any $\rho > 1$, there is no polynomial time $(\rho, 1)$ approximation for the (R-Upgrade-Total-Cost, Total Cost, Steiner Tree) problem even when restricted to bipartite graphs.

The above discussion leads us to state a general result for this case. Let Π be one of the unicriterion edge cost based problems (Total Cost, $\mathcal{S}$) considered in [AK+95, GW92, KV+94, BR+95].

Theorem 9.1 *Suppose there is a polynomial time ρ approximation algorithm for a problem Π. If the modified network contains a feasible subgraph in $\mathcal{S}$, then for all $\gamma > 0$, there is a polynomial time $((1 + 1/\gamma)\rho, (1 + \gamma)\rho)$ approximation algorithm for the* (R-Upgrade-Total-Cost, Total Cost, $\mathcal{S}$) *problem.* □

It can be seen that the above theorem can be generalized from the bicriteria case to the multicriteria case (with appropriate worsening of the performance guarantees).

10 Improved Algorithm for Treewidth Bounded Graphs and Linear Costs

In this section we will show how to obtain an improved algorithm for the class of treewidth bounded graphs when the reduction costs on the edges are *linear*. A class of treewidth-bounded graphs can be specified using a finite number of primitive graphs and a finite collection of binary composition rules. We use this characterization for proving our results. A class of treewidth-bounded graphs , is inductively defined as follows [BL+87].

1. The number of primitive graphs in , is finite.

2. Each graph in , has an ordered set of special nodes called **terminals**. The number of terminals in each graph is bounded by a constant, say k.

3. There is a finite collection of binary composition rules that operate only at terminals, either by identifying two terminals or adding an edge between terminals. A composition rule also determines the terminals of the resulting graph, which must be a subset of the terminals of the two graphs being composed.

The basic idea behind the algorithm in this section is to reduce the problem of improving the tree to some appropriately chosen bicriteria problem. To this end we recall the following result from [MRS$^+$95]:

Theorem 10.1 *1. There is a polynomial-time algorithm that, given an undirected graph G on n nodes with two nonnegative integral costs E and F on its edges, a bound $\mathcal{E}$, and a fixed $\gamma > 0$, constructs a spanning tree of G of total E-cost at most $(1+\gamma)\mathcal{E}$ and of total F-cost at most $(1+1/\gamma)$ times that of the minimum-F-cost of any spanning tree with total E-cost at most $\mathcal{E}$.*

2. For the class of treewidth-bounded graphs, there is a polynomial time algorithm that returns a spanning tree of total E-cost at most $\mathcal{E}$ and and of total F-cost at most $(1 + \varepsilon)$ times that of any spanning tree with total E-cost at most $\mathcal{E}$.

3. There is a polynomial-time algorithm that, given an undirected graph G on n nodes with two nonnegative integral costs E and F on its edges, a bound D, and a fixed

$\varepsilon > 0$, constructs a spanning tree of G of diameter at most $2\lceil \log_2 n \rceil D$ under the E-costs and of total F-cost at most $(1+\varepsilon)\lceil \log_2 n \rceil$ times that of the minimum-F-cost of any spanning tree with diameter at most D under E. □

We use the second part of the theorem to obtain an improved approximation for treewidth-bounded graphs under linear reduction costs as follows. First, we transform the original graph into another graph that can be fed into the algorithm from Theorem 10.1. To this end, we replace each edge $e = (u, v)$ of the original graph by a certain subgraph in such a way that the treewidth does not increase. The transformation procedure is shown in Figure 7 and an example of a transformation is displayed in Figure 8.

Let G be the original graph and G' be the graph obtained as a result of the transformation. Also, let $\mathrm{tw}(G)$ and $\mathrm{tw}(G')$ denote the treewidths of G and G' respectively. We have the following observation.

Observation 10.2 *Whenever $\mathrm{tw}(G) \geq 3$, we have that $\mathrm{tw}(G) = \mathrm{tw}(G')$.* □

TRANSFORM(ε)

1. **for** each edge $e = (u, v)$ in the graph let b_e be chosen so that $(1+\varepsilon)^{b_e} \leq \ell(e) - \ell^{\min}(e) \leq (1+\varepsilon)^{b_e+1}$.
2. Add $b_e + 2$ new vertices r_k, $k = -1, 0, \ldots, b_e$, which are joined together in a simple cycle.
3. **for all** $k, -1 \leq k \leq b_e$, join r_k to both u and v.
4. For $k \geq 0$, the edge (u, r_k) has E-cost $E(u, r_k) := \ell(e) - (1+\varepsilon)^k$ and F-cost $(1+\varepsilon)^k c_e$, while the edge (u, r_{-1}) has E-cost $\ell(e)$ and F-cost 0. All the edges (r_k, v) and (r_k, r_{k+1}) have their E-cost and F-cost set to zero.

   (An example of the above transformation for $b_e = 2$ can be seen in Figure 8).

Figure 7: Procedure used to transform G to G' in the approximation of (R-UPGRADE-TOTAL-COST, TOTAL COST, SPANNING TREE) on treewidth bounded graphs.

10.1 Correctness and Performance Guarantee

Let r^* denote the optimal reduction involving a cost of at most B, let T^* be a minimum spanning tree in $r^*(G)$ and let $L^* := \mathrm{MST}_G(\ell - r^*)$ be its weight in the modified graph. Also, let T' be a tree in G' with minimum total F-cost $F' := F(T')$ among all trees in G' that have E-cost at most ω. The performance guarantee provided by HEURISTIC-TW-UPGRADE shown in Figure 9 is summarized in the following theorem:

Theorem 10.3 *For the class of treewidth bounded graphs and linear reduction costs the following statement holds: For all fixed $\varepsilon > 0$,* HEURISTIC-TW-UPGRADE *is a $((1+\varepsilon), (1+\varepsilon))$-approximation algorithm for (*R-UPGRADE-TOTAL-COST, TOTAL COST, SPANNING TREE*).*

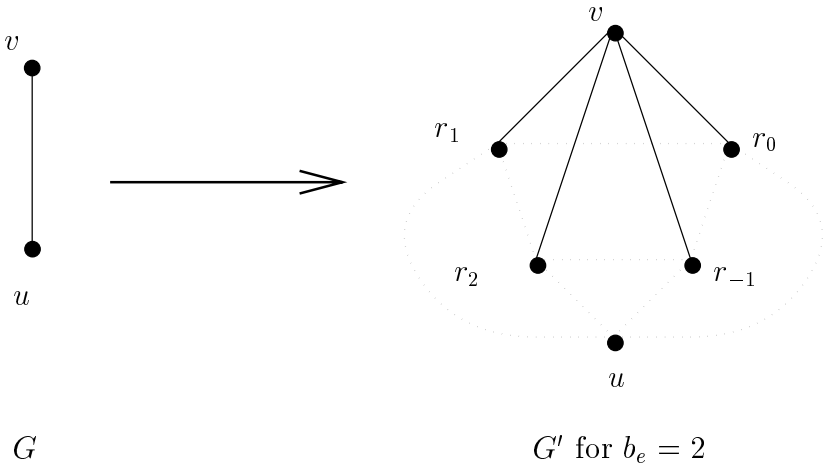


Figure 8: Example for the transformation on treewidth bounded graphs.

HEURISTIC-TW-UPGRADE(ε)

1 Call TRANSFORM(ε) to obtain a new graph G'.

2 Let $B' := (1+\varepsilon) \cdot B$

3 Use binary search to find the smallest integer $L' \in [(n-1)(\min_{e \in E} \ell^{\min}(e)), (n-1)(\max_{e \in E} \ell(e))$ such that the algorithm referred to in Part 2 of Theorem 10.1 called with the parameters L' for the E-cost bound $\mathcal{E}$ and $\varepsilon > 0$ returns a tree of F-cost at most B'.

4 Let T' be the tree generated by the algorithm from Theorem 10.1.

5 For each edge $e = (u, v)$, define the reduction r on e by $r(e) := 0$ if (u, r_{-1}) is included in T' and otherwise by $r(e) := (1+\varepsilon)^k$, where $0 \leq k \leq b_e$ is the minimum value such that (u, r_k) is included in T'.

6 **return** r.

Figure 9: Main Procedure for the approximation of (R-UPGRADE-TOTAL-COST, TOTAL COST, SPANNING TREE) on treewidth bounded graphs.

Proof: Let us first understand the relationship between B' and B and that between F' and L^*. Consider the tree T^* in G. We can define a tree T'' in G' in the following way: For an edge $e = (u, v) \in T^*$ that is reduced by $r^*(e)$ we select an edge (u, r_j) in G' of E-cost $\ell(e) - b(e)$, where $b(e)$ is selected in such a way that $\frac{b(e)}{(1+\varepsilon)} \leq r^*(e) \leq b(e)$. We also select the edge (r_j, v) to belong to T''. Observe that the edge (u, r_j) selected in the above fashion has its length reduced by at most $(1 + \varepsilon) \cdot r^*(e)$ and at least by $r^*(e)$. Using this fact the following claim can be proven.

Claim: The F-cost of the tree T'' is at most $(1 + \varepsilon)B$. The total E-cost of the tree T'' in G' is at most L^*. □

Hence we have demonstrated a witness tree T'' such that if the bound on the E-length is L^*, then the F-cost of this tree is bounded from above by $B' := (1 + \varepsilon)B$. Consequently, the *minimum* F-cost tree T' in G' (under the constraint that the E-cost does not exceed L^*) will have cost at most B'. Thus the binary search will terminate with a value $L' \leq L^*$.

Specifically, for our algorithm sketched above, the total weight $\mathrm{MST}_G(\ell - r)$, where r is the reduction returned by the heuristic, is then bounded from above by $(1+\varepsilon)L' \leq (1 + \varepsilon)L^*$. Moreover, the cost of reduction r which is defined in Step 5 of Heuristic-TW-Upgrade is no more than the F-cost of the tree T', which is found with the help of the algorithm from Theorem 10.1.

Since we know that the cost of this tree is bounded above by $B' = (1 + \varepsilon) \cdot B$ by the fact that the binary search has indeed terminated with some L', the claimed performance guarantee with respect to the budget follows. □

10.2 Running Time

We now show that the algorithm can be implemented to run in polynomial time. For this, observe that for a fixed value of $\varepsilon > 0$, the number of edges added (i.e., the value of b_e) is polynomial in the size of the input. This proves that the procedure Transform runs in time $\mathcal{O}(m \log M)$ where $M = \sum_{e \in E} \ell(e)$. Next, observe that the binary search in the main procedure can be done in polynomial time. Thus the algorithm can be executed in polynomial time.

10.3 Extensions and Related Remarks

In the following, we briefly outline the extensions of the above technique in solving other edge based network improvement problems.

First, by a slight extension of the ideas in [MRS+95] the above algorithm can be modified to obtain a $(1 + \epsilon, 1)$ approximation algorithm for (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) for treewidth-bounded graphs. The basic idea is to modify the algorithm in [MRS+95] to find a $(1 + \epsilon, 1)$ approximation algorithm to the bicriteria spanning tree problem. This combined with above procedure yields a $(1+\epsilon, 1)$ approximation algorithm for the (R-Upgrade-Total-Cost, Total Cost, Spanning Tree) problem when restricted to the class of treewidth bounded graphs.

Second, we note that using the same techniques as in the case of treewidth-bounded graphs and the Part 1 of Theorem 10.1, we can obtain a $((1 + 1/\gamma), (1 + \varepsilon)(1 + \gamma))$ approximation algorithm for the (R-UPGRADE-TOTAL-COST, TOTAL COST, SPANNING TREE) problem on general graphs with linear reduction costs, for any positive value of γ and ε. However, such an approximation algorithm would be inferior to the approximation algorithm (HEURISTIC-UPGRADE) given in Section 7 in the following ways.

1. Even if we ignore the results of Section 8, the running time of HEURISTIC-UPGRADE as stated in Section 7 is $\mathcal{O}(\log(\frac{nL_{max}}{\gamma\epsilon})T_{\text{MST}}(n, m))$ where L_{max} is the maximum length of an edge in G and $T_{\text{MST}}(n, m)$ is the time needed to compute a minimum spanning tree in a graph with n nodes and m edges. The approximation algorithm based on Part (1) of Theorem 10.1 would first construct (using the transformation shown in Figure 7) a graph G' by replacing each edge of G by a subgraph with $\Theta(\log B)$ edges and nodes. Thus, the resulting graph G' has $\Theta(n + m \log B)$ nodes and $\Theta(m \log B)$ edges. For this graph, as discussed in [MRS+95], the parametric search procedure would run in $\mathcal{O}(\log (L_{max}) T_{\text{MST}}(n + m \log B, m \log B))$ time. Using the best known value for $T_{\text{MST}}(n, m) = n + m \log \beta(m, n))$ [GGST86], it can be seen that HEURISTIC-UPGRADE is faster by a factor of $\mathcal{O}(\log B)$. This improvement in running time is particularly significant when the value of B is large. (For example, if $B = 2^{m^2}$, the time improvement factor is $\mathcal{O}(m^2)$.)

2. HEURISTIC-UPGRADE provides a performance guarantee of $(1 + 1/\gamma, 1 + \gamma)$ (ignoring the additive constant ε which can be made arbitrarily close to zero), thus improving the budget violation by the factor $(1 + \varepsilon)$.

3. As already mentioned, HEURISTIC-UPGRADE can handle a variety of different cost functions while the approximation algorithm based on Theorem 10.1 works only for linear cost functions.

4. HEURISTIC-UPGRADE does not require any additional space while the other approximation algorithm carries out a transformation that increases the size of the graph significantly.

Finally, the transformation of Figure 7 along with Part (3) of Theorem 10.1 can be used to obtain an $(\mathcal{O}(\log n), \mathcal{O}(\log n))$ approximation algorithm for the (R-UPGRADE-TOTAL-COST, DIAMETER, SPANNING TREE) problem and its variants (0/1 and integer reductions). The techniques immediately extend to the Steiner variants of the problems. The ideas are almost identical and hence we omit the proof.

11 Conclusions and Future Work

We studied the complexity and approximability of several natural network improvement problems. The results obtained in this paper are summarized in Table 1.

(Upgrade-Total-Cost, Total Cost, Steiner Tree) Problems			
	0/1	Integral	Rational
Trees	weakly NP-hard	weakly NP-hard Easy (linear c_e)	weakly NP-hard Easy (linear c_e)
Treewidth bounded Graphs	NP-hard $(1+\varepsilon, 1)$-approx.	NP-hard $(1+\varepsilon, 1)$-approx.	NP-hard $(1+\varepsilon, 1)$-approx.
General Graphs	strongly NP-hard	strongly NP-hard	strongly NP-hard
	hard to approx. within $(\rho, 1)$	hard to approx. within $(\rho, 1)$	hard to approx. within $(\rho, 1)$
	$(2(1+1/\gamma), 2(1+\varepsilon)(1+\gamma))$-approx.	$(2(1+1/\gamma), 2(1+\varepsilon)(1+\gamma))$-approx.	$(2(1+1/\gamma), 2(1+\gamma))$-approx.

Table 1: Approximation and Hardness Results for (Upgrade-Total-Cost, Total Cost, Steiner Tree) and related problems. Similar results hold for other general network design problems such as those considered in [GGP+94].

(Upgrade-Total-Cost, Diameter, Steiner Tree) Problems			
	0/1	Integral	Rational
Trees	weakly NP-hard	weakly NP-hard	weakly NP-hard
Treewidth bounded Graphs	NP-hard $(1+\varepsilon, 1)$-approx.	NP-hard $(1+\varepsilon, 1)$-approx.	NP-hard $(1+\varepsilon, 1)$-approx.
General Graphs	strongly NP-hard	strongly NP-hard	strongly NP-hard
	also hard to approx. within $(11/10-\varepsilon, (1-\varepsilon')\log n)$	also hard to approx. within $(11/10-\varepsilon, (1-\varepsilon')\log n)$	also hard to approx. within $(11/10-\varepsilon, (1-\varepsilon')\log n)$
	$(\mathcal{O}(\log n), \mathcal{O}(\log n))$-approx.	$(\mathcal{O}(\log n), \mathcal{O}(\log n))$-approx.	$(\mathcal{O}(\log n), \mathcal{O}(\log n))$-approx.

Table 2: Approximation and Hardness Results for (Upgrade-Total-Cost, Diameter, Steiner Tree) and related problems.

The results in this paper raise the following additional questions. One obvious open question is to improve the performance guarantees of the problems considered in this paper. Second, it is worth considering other related network improvement network design problems. As a step in this direction, in [KN+96], we have considered node-based network improvement problems and provided both hardness and easiness results for a number of such problems. Third, as a step further, it is interesting in general to look at other improvement problems for graphs. One such class of problems which might be interesting are the location theoretic problems such as the k-center problem. The paper by Berman [Ber92] represents the first work in this direction. Finally, it would be interesting to look at the above problems for special classes of graphs such as grid graphs, perfect graphs and investigate the existence of more efficient algorithms for the above problems restricted to these graph classes. As a step in this direction, in [KN+96], we show that several problems considered here have a fully polynomial approximation schemes (FPAS) when restricted to the class of treewidth bounded graphs.

Acknowledgements: It is a pleasure to acknowledge various constructive discussions with R. Ravi (Carnegie Mellon University) and Ravi Sundaram (Delta Trading Inc). Several results in this paper were developed as a result of extensive discussions with them. We thank Cynthia Phillips (Sandia National Laboratories) for useful conversations on related topics and pointers to literature. We also thank J. Plesnik for making available copies of his papers.

References

[AC+93] S. Arnborg, B. Courcelle, A. Proskurowski and D. Seese, "An Algebraic Theory of Graph Reductions," *Journal of the ACM (JACM)*, vol. 40:5, pp. 1134-1164 (1993).

[AK+95] A. Agrawal, P. Klein and R. Ravi, "When trees collide: an approximation algorithm for the generalized Steiner problem on networks," *SIAM Journal on Computing*, vol.24, pp. 440-456 (1995).

[AL+91] S. Arnborg, J. Lagergren and D. Seese, "Easy Problems for Tree-Decomposable Graphs," *Journal of Algorithms*, vol. 12, pp. 308-340 (1991).

[Ber92] O. Berman, *Improving the location of minisum facilities through network modification*, Annals of Operations Research **40** (1992), 1–16.

[BL+87] M.W. Bern, E.L. Lawler and A.L. Wong, "Linear -Time Computation of Optimal Subgraphs of Decomposable Graphs," *Journal of Algorithms*, vol. 8, pp. 216-235 (1987).

[BR+95] A. Blum, R. Ravi and S. Vempala, "A constant-factor approximation algorithm for the k-MST problem," To appear in the *Proceedings of the 28th Annual ACM Symposium on the Theory of Computation* (1996).

[Bo88] H.L. Bodlaender, "Dynamic programming on graphs of bounded treewidth," *Proceedings of the 15th International Colloquium on Automata Language and Programming*, LNCS vol. 317, pp. 105-118 (1988).

[CKR+92] J. Cong, A. B. Kahng, G. Robins, M. Sarafzadeh, and C. K. Wong, *Provably good performance driven global routing*, IEEE Transactions on Computer Aided Design **11** (1992), no. 6, 739–752.

[Col88] R. Cole, *Parallel merge sort*, SIAM Journal on Computing **17** (1988), no. 4, 770–785.

[FSO96] G. N. Frederickson and R. Solis-Oba, *Increasing the weight of minimum spanning tree*, Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96), January 1996, 539–546

[Fe95] U. Feige, "A threshold of $\ln n$ for approximating set cover," in the *Proceedings of the 28th Annual ACM Symposium on the Theory of Computation*, May 1996, 314–318.

[GGP+94] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, E. Tardos, and D. P. Williamson, *Improved approximation algorithms for network design problems*, Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94), January 1994, pp. 223–232.

[GGST86] H. N. Gabow, Z. Galil, T. H. Spencer, and R. E. Tarjan, *Efficient algorithms for finding minimum spanning trees in undirected and directed graphs*, Combinatorica **6** (1986), 109–122.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability (a guide to the theory of* NP*-completeness)*, W.H. Freeman and Company, San Francisco, CA, 1979.

[GW92] M. W. Goemans and D. P. Williamson, *A general approximation technique for constrained forest problems*, Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'92), January 1992, pp. 307–316. To appear in *SIAM Journal on Computing.*

[Has92] R. Hassin, *Approximation schemes for the restricted shortest path problem*, Mathematics of Operations Research **17** (1992), no. 1, 36–42.

[KJ83] B. Kadaba and J. Jaffe, *Routing to multiple destinations in computer networks*, IEEE Transactions on Communication **COM-31** (1983), 343–351.

[KV+94] S. Khuller and U. Vishkin, "Biconnectivity Approximations and Graph Carvings," *Journal of the ACM (JACM)*, vol. 41 , pp. 214-235, (1994).

[KN+96] S. O. Krumke, H. Noltemeier, M. V. Marathe, S. S. Ravi, R. Ravi and R. Sundaram, *On Optimal Strategies for Upgrading Networks*, July 1996, submitted for publication.

[KP95] D. Karger and S. Plotkin, *Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows*, Proceedings of the 27th Annual ACM Symposium on the Theory of Computing (STOC'95), May 1995, 18–25.

[KPP92a] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, *Multicasting for multimedia applications*, Proceedings of the IEEE INFOCOM'92, January 1992, 2078–2085.

[KPP93] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, *Multicast routing for multimedia communication*, IEEE/ACM Transactions on Networking **1** (1993), 286–292.

[LY94] C. Lund and M. Yannakakis, *On the hardness of approximating minimization problems*, Journal of the ACM **41** (1994), no. 5, 960–981.

[Meg83] N. Megiddo, *Applying parallel computation algorithms in the design of serial algorithms*, Journal of the ACM **30** (1983), no. 4, 852–865.

[MRS+95] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III, *Bicriteria network design problems*, Proceedings of the

22nd International Colloquium on Automata, Languages and Programming (ICALP'95), Lecture Notes in Computer Science, vol. 944, 1995, 487–498.

[Phi93] C. Phillips, *The network inhibition problem*, Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC'93), May 1993, 288–293.

[Pl81] J. Plesnik, *The complexity of designing a network with minimum diameter*, Networks **11** (1981), 77–85.

[Rav94] R. Ravi, *Rapid rumor ramification*, Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science (FOCS'94), November 1994, pp. 202–213.

[RMR+93] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III, *Many birds with one stone: Multi-objective approximation algorithms*, Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC'93), May 1993, 438–447.

[War92] A. Warburton, *Approximation of pareto optima in multiple-objective shortest path problems*, Operations Research **35** (1992), 70–79.

[ZPD94] Q. Zhu, M. Parsa, and W. Dai, *An iterative approach for delay bounded minimum Steiner tree construction*, Tech. Report UCSC-CRL-94-39, University of California, Santa Cruz, October 1994.

[Ze94] A. Z. Zelikovsky, *A 11/6-approximation algorithm for the Steiner problem on networks*, Algorithmica, **9** (1994), 463–470.